A Non-Adaptive Algorithm for the Quantitative Group Testing Problem

author names withheld

Editor: Under Review for COLT 2024

Abstract

Consider an n dimensional binary feature vector with sparsity k, i.e., at most k coordinates are equal to 1. The vector can be interpreted as the incident vector corresponding to n items out of which k items are *defective*. The *quantitative group testing* (QGT) problem aims at learning this binary feature vector by queries on subsets of the items that return the total number of defective items. We consider this problem under the *non-adaptive* scenario, that is, queries on subsets of items are designed collectively, allowing parallel execution. Most of the existing efficient non-adaptive algorithms for the sublinear regime where $k = n^{\alpha}$ with $0 < \alpha < 1$ fall short of the information-theoretic lower bound, with a multiplicative gap of $\log k$. Recently, Hahn-Klimroth and Müller (2022) closed this gap by by providing a non-adaptive algorithm with decoding complexity of $O(n^3)$. In this work, we present a concatenated construction method yielding a non-adaptive algorithm with the decoding complexity of $O(n^{2\alpha} + n\log^2 n)$. The probability of decoding failure is analyzed by establishing a connection between the QGT problem and the so-called *urn models*. Our algorithm reduces the gap between the information-theoretic and computational bound for the number of required queries/tests from $\log k$ to $\log\log k$. This contribution narrows the computational-semantic gap in the sublinear regime within decoding complexity $o(n^2)$, enhancing our understanding of computational complexity of non-adaptive algorithms for the QGT problem. Moreover, despite the $\log\log k$ gap in terms of the number of required tests, our algorithm is outperformed by the existing asymptotically optimal construction only when k is *exceptionally* large for moderate values of α, e.g., $k > 10^{27}$ for $\alpha = 0.7$, thereby highlighting the practical applicability of the proposed concatenated construction.

Keywords: Quantitative group testing, Statistical inference, Compressed sensing, Balls into bins problem

1. Introduction

Quantitative Group Testing (QGT) is the problem of detecting k defective items within a collection of n items through a series of tests conducted on m distinct pools. The result of each individual test yields the number of defective items present in the corresponding pool. This problem finds applications across a diverse spectrum of domains, including identifying rare variant carriers in genome sequencing Cao et al. (2014), network traffic control Wang et al. (2015), and resource allocation in random access channel De Marco et al. (2021), where the number of *defective items* can be estimated based on the collected information.

The QGT problem is often studied within two primary statistical models that define the arrangement of underlying items. In the *probabilistic* model, each item has a probability p of being defective, which is independent of other items. In the *combinatorial* model, the number of defective items, denoted as k, is known in advance. Strategies for addressing the QGT problem fall into two distinct categories: adaptive and non-adaptive. In the non-adaptive approach, all tests must be predetermined and may even be executed in parallel. In contrast, in the adaptive approach, one can

observe the results of previous tests and utilize this information to design subsequent tests. In this paper, we focus on the non-adaptive QGT problem within the combinatorial model.

Developing an algorithm for non-adaptive QGT can be considered as the construction of a binary matrix, where the rows represent the tests and the columns correspond to the items. An item is included in a pool test if the corresponding entry in the associated row is equal to 1. This setup bears a significant resemblance to the *compressive sensing* (CS) Candès et al. (2006); Donoho (2006) and *sparse recovery* problems that are studied extensively in the literature. In particular, QGT problem can be regraded as a *binary* CS problem , with the additional constraint that the measurement matrix must also be binary.

The QGT problem is often studied across various regimes of the underlying parameters k and n that denote the the number of defective items and the total number of items, respectively. Specifically, our focus lies on the so-called sublinear regime where $k = n^{\alpha}$ for some $0 < \alpha < 1$. The information-theoretic lowerbound on the number of tests required in a non-adaptive scheme for the QGT problem is

$$m_0 \stackrel{\text{def}}{=} \frac{2k}{\log k} \log \frac{n}{k}, \tag{1}$$

for the sublinear regime. Since this problem can be regarg=ded as a special case of CS problem, utilizing the linear programming methods proposed in this context recovers the underlying binary vector w.h.p. using $\Theta(k \ln n)$. Several studies in the literature addressed the construction of techniques tailored to the QGT problem, e.g., Soleymani et al. (2023); Gebhard et al. (2022); Feige and Lellouche (2020); Karimi et al. (2019). All such algorithms contribute to a reduction in the required number of tests compared to linear programming techniques, primarily focusing on improving multiplicative constant factors and the gap to the information theoretic gaps remains at $O(\log k)$.

Recently, Hahn-Klimroth and Müller (2022) successfully bridged this gap by introducing a non-adaptive algorithm that infers $\boldsymbol{x}$ w.h.p. with the number of tests approaching $\frac{1+\sqrt{\alpha}}{1-\sqrt{\alpha}} m_0$ as n grows large and a decoding complexity of $O(n^3)$. Our primary focus in this work lies on the decoding complexity of non-adaptive algorithms for the QGT problem. The key question is whether it is feasible to narrow the mentioned $O(\log k)$ gap with decoding complexity better $O(n^3)$, ideally approaching near-linear complexity. We provide a construction method that reduces the $O(\log k)$ gap to $O(\log \log k)$ in the number of required tests accompanied by a computational complexity of $O(n^{2\alpha} + n \log n)$. Our approach involves the introduction of a concatenation method that can be utilized to tackle the QGT problem through a divide-and-conquer strategy. The theoretical guarantee on the probability of successful detection of the defective items is established by introducing a certain probabilistic *urn model* Sprott (1978) and making a connection to the well-known *balls into bins* problem Park (1980). Our contribution can be summarized in the following:

- **Decoding Complexity:** In the sublinear regime, for $0 < \alpha < \frac{1}{2}$, the decoding complexity of our proposed algorithm is $O(n \log n)$, achieving the linearithmic decoding complexity while narrowing the computational-semantic gap from $O(\log k)$ to $O(\log \log k)$ in number of tests. For $\frac{1}{2} < \alpha < 1$, the complexity is $o(n^{2\alpha})$.
- **Non-asymptotic behavior:** Our proposed algorithm also outperforms the algorithm proposed by Hahn-Klimroth and Müller (2022) in the *non-asymptotic* regime in terms of the number of tests for *moderate* values of α. In other words, in order for the mentioned algorithm to outperform ours in terms of the number of tests, the parameter n must be *superficially large* for the *moderate* values of α.

The rest of the paper is organized as follows. Section 2 introduces the system model and presents a concatenation approach that is utilized in our proposed test matrix design Section 3 offers an overview of well-established findings pertaining to the *balls into bins* problem and introduces a novel urn model. We establish a connection between these two models, enabling us to employ the probability bounds from the former for the latter. Finally, in Section 4, we revisit two fundamental components that can serve as the building blocks for our concatenated scheme, and provide our construction and analyse its performance.

2. Problem Setting and a Concatenated Construction

2.1. Notation

The vectors and matrices are represented by bold lower and upper case characters, respectively. The ith component of a vector $\boldsymbol{a}$ is represented by a_i. The number of non-zero elements in $\boldsymbol{a}$ is referred to as the Hamming weight of $\boldsymbol{a}$ and is denoted by $|\boldsymbol{a}|$. The Hamming distance between two vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ is $|\boldsymbol{a}-\boldsymbol{b}|$. In this draft, log denotes the base 2 logarithm, while ln represents the natural logarithm with base e.

2.2. Problem Setting

The problem of quantitative group testing (QGT) is the following: A set of n items is given out of which k items are defective. The *incident* vector corresponding to these items is a binary vector $\boldsymbol{x} \in \{0,1\}^n$ such that $\boldsymbol{x}_i = 1$ if the item is defective and $\boldsymbol{x}_i = 0$, otherwise. This problem is reffered to as (n,k)-QGT problem. The goal is to identify all defective items, i.e., recover $\boldsymbol{x}$, by performing tests/measurements over as few subsets of the items as possible. The *outcome* of a test is the number of defective items belonging to the underlying subset. This problem is also referred to as the *coin weighing problem* in the literature where one attempts to detect k counterfeit coins from a collection of n coins using a *spring scale*, given that all counterfeit coins weight the same, which is different from the weight of genuine coins. Designing a non-adaptive strategy for the (n,k)-QGT problem is equivalent to constructing a binary matrix with n columns such that the sum of any subset of size at most k of the columns is distinct. Such matrix is referred to as a *search matrix* which is defined next in definition 1. Let $\mathscr{B}_{n,k}$ denote the set of all binary vectors of size n that have at most k non-zero elements.

Definition 1 *A binary matrix $\boldsymbol{A} \in \{0,1\}^{m\times n}$ is called an (n,k)-search matrix if $\boldsymbol{Ax}$ is distinct for all $\boldsymbol{x} \in \mathscr{B}_{n,k}$, i.e.,*

$$\boldsymbol{Ax} \neq \boldsymbol{Ax}'$$

for all $\boldsymbol{x} \neq \boldsymbol{x}' \in \mathscr{B}_{n,k}$. In particular, if $n = k$, the matrix is referred to as a search matrix.

In the context of the (n,k)-QGT problem, the objective is to minimize the number of rows needed in the construction of a k-search matrix.

Next, we extend the definition of search matrix to the case where the entries of $\boldsymbol{x}$ are non-negative integers.

Definition 2 *A binary matrix $\boldsymbol{D} \in \{0,1\}^{m\times n}$ is called a $(d_1,\cdots,d_n)$-detecting matrix if $\boldsymbol{Dx}$ is distinct for all $\boldsymbol{x} \in \{\boldsymbol{a} : 0 \leqslant \boldsymbol{a}_i < d_i, \forall i \in [n]\}$, i.e.,*

$$\boldsymbol{Dx} \neq \boldsymbol{Dx}', \qquad \forall \boldsymbol{x} \neq \boldsymbol{x}'.$$

In particular, if $d_i = d$ for all $i \in [n]$, the matrix is referred to as an (n, d)-detecting matrix.

We leverage an off-the-shelf construction for the (n, d)-GQGT problem, in particular the one introduced in Bshouty (2009). This serves as a building block to arrive at measurement matrices for the QGT problem if the underlying incident vector $\boldsymbol{x}$ is a *balanced* vector. The notion of balanced vector is defined next.

Definition 3 *A binary vector $\boldsymbol{x}$ of length n is called an (m, t)-balanced vector for some integer m that divides n if*

$$\max\Big(\sum_{i=1}^{\frac{n}{m}} x_i, \sum_{i=\frac{n}{m}+1}^{2\frac{n}{m}} x_i, \cdots, \sum_{i=n-\frac{n}{m}+1}^{n} x_i\Big) < t.$$

In other words, in an (m, t)-balanced vector $\boldsymbol{x}$, the number of ones inside each block of length $\frac{n}{m}$ does not exceed t.

2.3. Concatenated Construction

One of the main ideas of the construction provided in this paper involves *reducing* a QGT problem into carefully selected *smaller* QGT subproblems. These subproblems are then tackled using a test matrix tailored for solving the smaller instances, in conjunction with a detection matrix with precisely crafted parameters. To establish this, the following theorem demonstrates that the Kronecker of an (n_1, d)-detecting matrix with an (n_2, d)-search matrix can uniquely recover any vector $\boldsymbol{x}$ of length $n_1 n_2$ that is (n_1, d)-balanced.

Theorem 4 *Let $\boldsymbol{A} \in \{0,1\}^{m_1 \times n_1}$ and $\boldsymbol{B} \in \{0,1\}^{m_2 \times n_2}$ be an (n_1, d)-detecting matrix and an (n_2, d)-search matrix, respectively. Let*

$$\boldsymbol{C} \stackrel{\text{def}}{=} \boldsymbol{A} \otimes \boldsymbol{B}.$$

Then, one can uniquely recover the binary vector $\boldsymbol{x}$ of length $n_1 n_2$ from $\boldsymbol{Cx}$ if $\boldsymbol{x}$ is (n_1, d)-balanced.

Moreover, if there exist algorithms $\mathscr{A}$ and $\mathscr{B}$ that retrieve $\boldsymbol{a}$ and $\boldsymbol{a}'$ from $\boldsymbol{Aa}$ and $\boldsymbol{Ba}'$ within time complexities of $O_{\mathscr{A}}$ and $O_{\mathscr{B}}$, respectively, then one can recover $\boldsymbol{x}$ from $\boldsymbol{Cx}$ with a combined time complexity of $m_2 O_{\mathscr{A}} + n_1 O_{\mathscr{B}}$.

Proof: Suppose $\boldsymbol{x}$ is a binary vector of length $n_1 n_2$ that is (n_1, d)-balanced. That is, we have $\sum_{i=(j-1)n_2+1}^{jn_2} x_i < d$ for all $j \in [n_1]$. We prove the statement of the theorem by proposing a decoder that recovers $\boldsymbol{x}$ from $\boldsymbol{Cx}$. Suppose that there exist algorithms $\mathscr{A} : \mathbb{N}_0^{m_1} \to \{0, 1, \cdots, d\}^{n_1}$ and $\mathscr{B} : \mathbb{N}_0^{m_2} \to \{0,1\}^{n_2}$ that recover $\boldsymbol{a}$ and $\boldsymbol{a}'$ from $\boldsymbol{Aa}$ and $\boldsymbol{Ba}'$ within time complexities of O_1 and O_2, respectively. Note that a straightforward exhaustive search can always recover the corresponding vectors $\boldsymbol{a}$ and $\boldsymbol{a}'$, although with an exponential time complexity. This is due to the fact that $\boldsymbol{A}$ is an (n_1, d)-detecting matrix and $\boldsymbol{B}$ is an (n_2, d)-search matrix. Hence such decoding algorithms always exist. Let $\boldsymbol{b} \stackrel{\text{def}}{=} \boldsymbol{Cx}$ denote the corresponding measurement vector. We use the so-called mixed Kronecker matrix-vector product property as follows:

$$\operatorname{vec}(\boldsymbol{BXA}^T) = (\boldsymbol{A} \otimes \boldsymbol{B}) \operatorname{vec}(\boldsymbol{X}), \tag{2}$$

for any $n_2 \times n_1$ matrix $\boldsymbol{X}$, where the vec$(\cdot)$ operator constructs a column vector from the input matrix $\boldsymbol{T} = [\boldsymbol{t}_1|\cdots|\boldsymbol{t}_{m_1}]$ by vertically stacking the column vectors of $\boldsymbol{T}$ beneath each other, i.e., $\text{vec}(\boldsymbol{T}) \stackrel{\text{def}}{=} [\boldsymbol{t}_1^t, \cdots, \boldsymbol{t}_{m_1}^t]^t$. We partition $\boldsymbol{x}$ into n_1 column vectors of length n_2 and arrange them as the columns of a new $n_2 \times n_1$ matrix referred to as $\boldsymbol{X}$, i.e., $\boldsymbol{X} \stackrel{\text{def}}{=} [\boldsymbol{x}_1|\cdots|\boldsymbol{x}_{n_1}]$. Following this, it can be verified that $\text{vec}(\boldsymbol{X}) = \boldsymbol{x}$. Then, one can write

$$\boldsymbol{F}_{m_2\times n_1} \stackrel{\text{def}}{=} \boldsymbol{B}_{m_2\times n_2}\boldsymbol{X}_{n_2\times n_1} = [\boldsymbol{B}\boldsymbol{x}_1|\boldsymbol{B}\boldsymbol{x}_2|\cdots|\boldsymbol{B}\boldsymbol{x}_{n_1}]. \tag{3}$$

Let $\boldsymbol{F}^T_{n_1\times m_2} = [\tilde{\boldsymbol{f}}_1|\tilde{\boldsymbol{f}}_2|\cdots|\tilde{\boldsymbol{f}}_{m_2}]$. Note that $\tilde{\boldsymbol{f}}_i$'s represent the columns of $\boldsymbol{F}^T$, hence they are column vectors of length n_1. Then, we have

$$\begin{aligned}\text{vec}(\boldsymbol{B}\boldsymbol{X}\boldsymbol{A}^T) &= \text{vec}(\boldsymbol{F}\boldsymbol{A}^T) = \text{vec}((\boldsymbol{A}\boldsymbol{F}^T)^T) = \text{vec}([\boldsymbol{A}\tilde{\boldsymbol{f}}_1|\boldsymbol{A}\tilde{\boldsymbol{f}}_2|\cdots|\boldsymbol{A}\tilde{\boldsymbol{f}}_{m_2}]^T) &(4)\\ &= \text{vec}([\boldsymbol{y}_1|\cdots|\boldsymbol{y}_{m_1}]^T) = [y_{11},\cdots,y_{m_2 1}, y_{12},\cdots,y_{m_2 2},\cdots,y_{1m_1},\cdots,y_{m_2 m_1}]^T, &(5)\end{aligned}$$

where $\boldsymbol{y}_i = [y_{i1}, y_{i2},\cdots,y_{im_1}]^T \stackrel{\text{def}}{=} \boldsymbol{A}\tilde{\boldsymbol{f}}_i$, for all $i \in [m_2]$. Combining (2) together with (5) and recalling that $\boldsymbol{A}\otimes\boldsymbol{B} = \boldsymbol{C}$ and $\text{vec}(\boldsymbol{X}) = \boldsymbol{x}$ result in

$$\boldsymbol{b} = [y_{11},\cdots,y_{m_2 1}, y_{12},\cdots,y_{m_2 2},\cdots,y_{1m_1},\cdots,y_{m_2 m_1}]^T. \tag{6}$$

Hence, all $\boldsymbol{y}_i$ values for each $i \in [m_2]$ can be deduced from the measurement vector $\boldsymbol{b}$ using (6); specifically, $y_{ij} = b_{(j-1)m_2+i}$ for all $i \in [m_2]$ and $j \in [m_1]$. Then, one can arrange the following equations:

$$\boldsymbol{A}\tilde{\boldsymbol{f}}_i = \boldsymbol{y}_i, \qquad \forall i \in [m_2]. \tag{7}$$

Note that all $\tilde{\boldsymbol{f}}_i$'s then can be determined by utilizing the decoding algorithm corresponding to the matrix $\boldsymbol{A}$ which is algorithm $\mathscr{A}$, i.e., $\tilde{\boldsymbol{f}}_i = \mathscr{A}(\boldsymbol{y}_i)$, provided that the entries of $\tilde{\boldsymbol{f}}_i$'s are not larger than d. This requirement is necessary because matrix $\mathbf{A}$ is an (n_1, d)-detecting matrix. Hence the correct recovery of $\tilde{\boldsymbol{f}}_i$ is only guaranteed if all the entries of $\tilde{\boldsymbol{f}}_i$ do not exceed d. As $\tilde{\boldsymbol{f}}_i$'s are the columns of $\boldsymbol{F}^T$, it is sufficient to show that the entries of $\boldsymbol{F}$ do not exceed d. To establish this, note that all entries of $\boldsymbol{B}\boldsymbol{x}_j$ for all $j \in [n_1]$ are not greater than d. This follows because each entry is the sum of at most n_2 terms binary variables (recall that $\boldsymbol{B}$ is a binary matrix with n_2 columns), out of which at most d of them are 1. The latter holds since $\boldsymbol{x}$ is (n_1, d) balanced, leading to the constraint that the total number of 1's in $\boldsymbol{x}_j$ does not exceed d, according to Definition 3. Therefore, the entries of $\mathbf{F}$ satisfy the same condition due to (3). This confirms that none of the entries in $\tilde{\boldsymbol{f}}_i$ exceed d. As a result, one can reconstruct the matrix $\boldsymbol{F}$ by invoking the algorithm $\mathscr{A}$ for m_2 times and stacking the results as the columns of $\boldsymbol{F}^T$. Next, let $\boldsymbol{f}_j$ for all $j \in [n_1]$ denote the columns of $\boldsymbol{F}$, i.e., $\boldsymbol{F}_{m_2\times n_1} = [\boldsymbol{f}_1|\boldsymbol{f}_2|\cdots|\boldsymbol{f}_{n_1}]$. This together with (3) leads to

$$\boldsymbol{B}\boldsymbol{x}_j = \boldsymbol{f}_j, \qquad \forall j \in [n_1]. \tag{8}$$

Similarly, each of the individual equations can be solved by invoking the decoding algorithm corresponding to the test matrix $\boldsymbol{B}$, namely, $\mathscr{B}$. In other words, we have $\boldsymbol{x}_j = \mathscr{B}(\boldsymbol{f}_j)$ for all $j \in [n_1]$. Therefore, the incident vector $\boldsymbol{x}$ can be recovered by invoking the algorithms $\mathscr{A}$ and $\mathscr{B}$ for m_2 and n_1 times, respectively. This results in $m_2 O_1 + n_1 O_2$ combined time complexity to decode $\boldsymbol{x}$ from $\boldsymbol{C}\boldsymbol{x}$, which completes the proof. ■

The result of Theorem 4 implies that when dealing with a binary vector $\boldsymbol{x}$ of length $n_1 n_2$ that is (n_1, d)-balanced, the Kronecker product of an (n_1, d)-detection matrix with an (n_2, d)-search matrix yields a test matrix that can uniquely recover $\boldsymbol{x}$. In order to gain a deeper insight into this construction, lets regard the problem as a coin weighing problem with $n_1 n_2$ coins. The (n_1, d)-balanced assumption implies that one can partition all coins into n_1 subgroups consisting of n_2 coins, such that the total number of counterfeit coins (corresponding to 1's in the underlying incident vector $\boldsymbol{x}$) inside each subgroup does not exceed d. Suppose that the matrix $\boldsymbol{B} \in \{0,1\}^{m_2 \times n_2}$ is an (n_2, d)-search matrix which solves this sub-problem. A naive approach then would be to utilize $\boldsymbol{B}$ for each subgroup independently in order to identify the location of all counterfeit coins. The overall test matrix can be understood as $\boldsymbol{I}_{n_1} \otimes \boldsymbol{B}$, which results in $m_2 n_1$ total number of tests. However, the test matrix constructed in Theorem 4, which is the concatenation of $\boldsymbol{B}$ with an (n_1, d)-detecting matrix, requires only $m_2 m_1$ tests where $m_1 = o(n_1)$ is achievable. Therefore, the result of Theorem 4 introduces an approach for addressing the QGT problem through a divide-and-conquer strategy. However, this is true only when the underlying binary vector to be reconstructed adheres to a specific condition termed as a *balanced vector*, as defined in Definition 3. We will show in the next section that this holds with probability approaching 1 as the length of $\boldsymbol{x}$ grows large, if one carefully chooses the size of the underlying subgroups under the combinatorial model for the QGT problem.

3. Connection to Urn Models

In this section we review some results on the so-called *balls into bins* problem from the literature. We then demonstrate that a randomly selected vector $\boldsymbol{x}$ from the set $\mathcal{B}_{n,k}$ is a balanced vector with certain parameters by making a connection between the underlying probabilistic model for the incident vector $\boldsymbol{x}$ to the balls into bins problem. The description of this problem is as follows: There are n bins and m balls. Each ball is thrown into one of these n bins. In the simplest scenario, the bins are chosen uniformly at random, i.e., each with probability $\frac{1}{n}$, as illustrated in Figure 1. There are various questions that arise concerning the statistics of the ball distributions including the statistics of the number of empty bins, the maximum number of balls inside a single bins, etc. Such problems have applications in computer science and engineering such as online load balancing and analysis of the hashing algorithms. Several variants of this problem are extensively studied in the literature, collectively falling under the broad category referred to as *urn models* Sprott (1978); Park (1980). In particular, our focus lies on the statistical analysis of specific a parameter within this framework. Let M denote the maximum number of balls inside the individual bins in the balls into bins problem. We will use the following result on the probability of M exceeding a certain threshold from Raab and Steger (1998).

Theorem 5 (Raab and Steger (1998), Theorem 1) *Let M be the random variable that counts the maximum number of balls in any bin, if we throw m balls independently and uniformly at random into n bins. Then, if $m = cn \ln n$ for some constant c, we have*

$$\Pr[M > (d_c - 1 + \alpha) \ln n] = o(1), \tag{9}$$

for any $\alpha > 1$. Here d_c is a solution to

$$1 + x(\ln c - \ln x + 1) - c = 0 \tag{10}$$

that is larger than c.

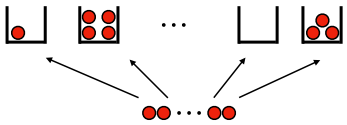

Figure 1: Illustration of balls into bins problem with m balls and n bins.

Figure 2: Demonstration of sampling without replacement as an LDLC model. n items are divided into l and each time one balls is colored red that represents a defective item.

It's worth noting that within the parameter range examined in Theorem 5, on average, each bin holds around $c \ln n$ balls. Notably, the pigeonhole principle implies that the maximum is always greater than $c \ln n$. The significance of the result in Theorem 5 lies in the assurance that the maximum number of the balls contained in each bin deviates from its average only by a *constant* that does not deepened on n.

Next, we explore the connection between the (n, k)-QGT problem with the balls into bins problem and demonstrate that how the result of Theorem 5 can be applied within the context of the QGT problem. In this paper, we consider the combinatorial model for the QGT problem, wherein the total number of defective items is known a priori. It is permissible to assume, without loss of generality, that the subset of defective items is uniformly distributed over all k-subsets of $[n]$. This is because one can always randomly shuffle the incident vector $\boldsymbol{x}$, thereby resulting in a uniform distribution for the shuffled vector. From the prospective of the entropy of $\boldsymbol{x}$, the uniform distribution corresponds to the maximum possible entropy for $\boldsymbol{x}$. Consequently, an algorithm capable of recovering $\boldsymbol{x}$ under the uniform prior assumption with a certain probability can similarly decode $\boldsymbol{x}$ under *any* other prior with an equivalent or even better probability of success. Roughly speaking, the uniform prior represents the most challenging scenario within the (n, k)-QGT problem. This assertion can also be inferred by recognizing that random shuffling never reduces the entropy of $\boldsymbol{x}$. Therefore, for the rest of this paper, we always assume that $\boldsymbol{x}$ is uniformly distributed over $\mathscr{B}_{n,k}$.

One possible approach to generate $\boldsymbol{x}$ according to the uniform distribution from $\mathscr{B}_{n,k}$ is to repeat the following for k steps: At each step, one chooses an item among those that have not been chosen in previous steps uniformly at random and labels it as a defective item. This is often referred to as *sampling without replacement* in the literature. Let the set of all items be partitioned into l distinct subsets, as depicted in Figure 2. Each of these partitions can be viewed as a bin, and the act of selecting an item from a particular partition can be regarded as placing a *ball* into that bin. Therefore, sampling k items without replacement from the set of n items can can be comprehended within the framework of urn models. One may note that the procedure of placing balls in the mentioned urn model does not align with the procedure in the balls into bins problem, where the result of Theorem 5 remains applicable. To see that, let X_{ij} for $i \in [l]$ and $j \in [k]$ denote random variables representing the count of items that have not been selected by the end of step j. The main difference of the above urn model with the balls into bins problem is that the bin i is selected at step j with a probability that is proportional to X_{ij}, whereas all bins are chosen uniformly at random in all steps in the balls into bins problem. Thus, the urn model depicted in Figure 2 differs from the balls into bins problem in two major aspects. First, the *capacity* of each urn is *limited*, meaning there is a constraint on the number of balls that can be placed into each bin. In particular, for the scenario considered above where n items are divided into l partitions, the number of defective items in each bin can not exceed $\frac{n}{l}$. Second, when regarded as an urn model, sampling without

replacement intrinsically exhibits a form of negative reinforcement, where a newly selected ball is less likely to be placed in a bin that already contains more balls, compared to other bins. We refer to this urn model as *linearly de-preferential with limited capacity* (LDLC) urn model. In the next lemma, we show that the probability of the maximum number of balls inside bins in the LDLC problem is always lower bounded by that of the balls into bins problem.

Lemma 6 *Let $M^{\mathscr{B}}_{n,m}$ denote the maximum number of balls in the balls into bins problem with n bins and m balls. Let also $M^{\mathscr{L}}_{n,m}$ denote the maximum number of balls in the LDLC urn model with n bins and m balls. Then, for any positive integer t, we have*

$$\Pr[M^{\mathscr{B}}_{n,m} \leqslant t] \leqslant \Pr[M^{\mathscr{L}}_{n,m} \leqslant t]. \tag{11}$$

Proof: The proof is moved to Appendix A due to space limitations. ■

The result of Lemma 6 can be intuitively justified by comparing the bin selection process in each step. Note that in the balls into bins problem, a ball is placed into a randomly chosen bin, regardless of the arrangement of previously placed balls. In contrast, in the LDLC problem, the probability of selecting a bin with a higher number of balls already in it is reduced, creating a disincentive to choose bins with more balls. This *negative feedback mechanism* makes it less likely for the bin containing the maximum number of balls to be chosen for the next ball placement. In other words, the random variable $M^{\mathscr{L}}_{n,m}$ is *less likely* to increase compared to $M^{\mathscr{B}}_{n,m}$ at each step.

Building upon the results of Theorem 5 and Lemma 6, it is shown in the following corollary that the incident vector $\boldsymbol{x}$ drawn uniformly from $\mathscr{B}_{n,k}$ is and $(l, c' \log k)$-balanced vector with high probability for any constant $c' > 2$ and a carefully chosen parameter l.

Corollary 7 *Let $\boldsymbol{x}$ be a vector that is drawn from $\mathscr{B}_{n,k}$ according to a uniform distribution. Then, $\boldsymbol{x}$ is $(\frac{k}{\ln k}, (e+\gamma)\ln k)$-balanced with probability $1 - o(1)$, for any constant $\gamma > 0$.*

Proof: Lets partition $\boldsymbol{x}$ into l distinct subsets as shown in Figure 2 where $l = \frac{k}{\ln k}$. Note that this resembles an LDLC urn model with $m = k$ balls and $n = \frac{k}{\ln k}$ bins. Then,

$$\Pr[M^{\mathscr{L}}_{\frac{k}{\ln k},k} \leqslant (e+\gamma)\ln k] \leqslant \Pr[M^{\mathscr{B}}_{\frac{k}{\log k},k} \leqslant (e+\gamma)\ln k] \tag{12}$$

$$= 1 - o(1), \tag{13}$$

where (12) follows by the result of Lemma 6 and (13) is established by recognizing that the number of balls and bins satisfy the condition in Theorem 5. In other words, we have $n \ln n = \frac{k}{\ln k} \log(\frac{k}{\ln k}) = k(1 + o(1)) = m$, corresponding to the choice of $c = 1$ in Theorem 5. Consequently, d_c represents the larger solution to the equation (10) with $c = 1$, which leads to $d_c = e$. Lastly, note that $\alpha - 1$ for any $\alpha > 1$ can be replaced with γ for any $\gamma > 0$. ■

The implication of the result established in Corollary 7 bears considerable significance for the (n,k)-QGT problem. In essence, as n grows large, this implies that the problem can be decomposed into $\frac{k}{\ln k}$ instances of $(\frac{n \ln k}{k}, e \ln k)$-QGT problems. This decomposition enables us to tackle the problem by using the concatenated approach detailed in Section 2. Roughly speaking, this suggests that a *typical* incident vector $\boldsymbol{x}$, selected uniformly at random from $\mathscr{B}_{n,k}$, can be divided into roughly $\frac{k}{\ln k}$ distinct subsets, with the total count of defective items across all these subsets not exceeding $e \ln k$. Note that the pigeonhole principle implies that the maximum is at least $\log k$. This shows the

bound derived through observing the connection to the balls into bins problem, as characterized in Lemma 6, is essentially optimal up to a constant factor less than 2. In the next section, we will provide specific details about the constructions employed for certain detecting and search matrices that are utilized in the concatenation method proposed in Section 2 to tackle the (n,k)-QGT problem.

4. Proposed Construction

In this section we briefly review specific constructions for d-detecting matrices and (n,k)-search matrices that are utilized in our construction. Then, building upon the results provided in Section 2 and Section 3, we propose a concatenated construction that outperforms the state-of-the-art constructions in the literature for the QGT problem in the sublinear regime. First, we consider an optimal construction for d-detecting matrix from the literature.

4.1. An Optimal Construction for Detecting Matrices

We provide a concise summary of the construction method proposed in Bshouty (2009) for detecting matrices. This construction is designed to generate optimal $(d_1,\cdots,d_n)$-detecting matrices. However, our specific focus is on detecting matrices where $d_i = d$ for all $i \in [n]$. Therefore, we will explore the construction tailored to this particular case.

The primary idea behind constructing the (n,d)-detection matrix in Bshouty (2009) involves carefully crafting a certain collection of binary functions defined on the domain $\{-1,+1\}^{2^\nu}$ for a given integer ν. The specific characteristics of these functions are detailed in Appendix B. These functions are labeled by $\boldsymbol{a} \in \{0,1\}^\nu$ and $i \in [l_{\boldsymbol{a}}]$, where $l_{\boldsymbol{a}}$ is a non-negative integer that satisfies the condition $d^{l_{\boldsymbol{a}}} \leqslant 2^{|\boldsymbol{a}|-1} < d^{l_{\boldsymbol{a}}+1}$. We represent these functions as $g_{\boldsymbol{a},l_{\boldsymbol{a}}} : \{-1,+1\}^\nu \to \{0,1\}$, and the entire set containing such functions is denoted as $\mathcal{G}_{\nu,d}$. The d-detecting matrix takes the form of a matrix with rows labeled by binary vectors of ± 1 values and columns labeled by functions from the set $\mathcal{G}_{\nu,d}$. Then, the (i,j) entry of $\boldsymbol{M}$ is equal to the evaluation of the function corresponding to column j, evaluated at the evaluation point corresponding to row i. In simpler terms, each column of this matrix stores the results of applying a function from $\mathcal{G}_{\nu,d}$ to all possible inputs from $\{-1,+1\}^\nu$. The number of rows in matrix $\boldsymbol{M}$ is at most

$$\frac{2n}{\log n}\log d\,(1+o(1)), \tag{14}$$

which asymptotically matches the information theoretic lower bound for the minimum number of non-adaptive tests. Let $B \stackrel{\text{def}}{=} \{\chi_{\boldsymbol{a}}(\boldsymbol{x}) \stackrel{\text{def}}{=} \prod_{a_i=1} x_i | \boldsymbol{a} \in \{0,1\}^\nu\}$. Since B is an orthogonal set of functions, any function $f(x)$ with domain $\{-1,1\}^\nu$, including the functions in $\mathcal{G}_{\nu,d}$, can be uniquely represented as

$$f(x) = \sum_{\boldsymbol{a}\in\{0,1\}^\nu} \hat{f}_{\boldsymbol{a}}\chi_{\boldsymbol{a}}(x), \tag{15}$$

where $\hat{f}_{\boldsymbol{a}}$ is called the Fourier coefficient of $\chi_{\boldsymbol{a}}(x)$ in $f(x)$ and is equal to

$$\hat{f}_{\boldsymbol{a}} = \frac{1}{2^\nu}\sum_{\boldsymbol{x}\in\{-1,+1\}^\nu} f(\boldsymbol{x})\chi_{\boldsymbol{a}}(\boldsymbol{x}). \tag{16}$$

The functions in the set $\mathcal{G}_{\nu,d}$ have an important property that is utilized in establishing the matrix $\boldsymbol{M}$ as a d-detecting matrix. This property also enables the development of an efficient decoder for

determining $\boldsymbol{x}$ from $\boldsymbol{Mx}$, provided that all entries of $\boldsymbol{x}$ are non-negative integers less than d. This property can be described as in the following. Consider a linear combination of such functions as $h_{\boldsymbol{\lambda}}(x) = \sum_{\boldsymbol{a}\in A} \sum_{i\in\{0,1,\cdots,l_{\boldsymbol{a}}\}} \lambda_{\boldsymbol{a},i} g_{\boldsymbol{a},i}(x)$ for some $A \subset \{0,1\}^{\nu}$. Then, if $\mathbf{b} \in A$ is a maximal element in A, the Fourier coefficient of $h(x)$ in $\chi_{\mathbf{b}}$ is equal to $\sum_{j\in\{0,1,\cdots,l_{\boldsymbol{a}}\}} \lambda_{\mathbf{b},j} d^j$. This property is exploited during the decoding procedure as discussed below.

Suppose we have a vector $\boldsymbol{\lambda} \in \{0,1,\cdots,d-1\}^n$. This vector can be regarded as a function within the basis $\mathcal{G}_{\nu,d}$. Therefore, recovering $\boldsymbol{\lambda}$ is equivalent to retrieving the function $h_{\lambda}(x) = \sum_{\boldsymbol{a}\in\{0,1\}^{\nu}} \sum_{i\in\{0,1,\cdots,l_{\boldsymbol{a}}\}} \lambda_{\boldsymbol{a},i} g_{\boldsymbol{a},i}(x)$. Note that the measurement vector $\boldsymbol{M\lambda}$ provides all the evaluations of $h_{\lambda}(x)$ over $\{-1,1\}^{\nu}$. Therefore one can determine all the Fourier coefficients of $h_{\lambda}(x)$ at the decoder and search for a maximal $\boldsymbol{a} \in \{0,1\}^{\nu}$ whose corresponding Fourier coefficient is non-zero. All the entries of $\boldsymbol{\lambda}$ corresponding to $\boldsymbol{a}$ can be recovered from the expansion of $\sum_{j\in 0,1,\cdots,l_{\boldsymbol{a}}} \lambda_{\mathbf{a},j} d^j$ in base d. This process can be repeated recursively, replacing $h_{\lambda}(x)$ with $h_{\lambda}(x) - \sum_{j\in 0,1,\cdots,l_{\boldsymbol{a}}} \lambda_{\mathbf{a},j} g_{\boldsymbol{a},j}$ until all the entries of $\boldsymbol{\lambda}$ are determined. This demonstrates that the recovery of $\boldsymbol{\lambda}$ can be accomplished in $O(n^2)$ time. Next, we present our observation that extends the decoding algorithm to handle cases where the vector $\boldsymbol{\lambda}$ violates the assumption that all of its non-negative integer entries are less than d. In particular, we show that the decoding procedure discussed earlier, with a slight adjustment, can determine whether any entry of λ exceeds d and declare it as a decoding failure. This modified algorithm is detailed in Algorithm 1.

Algorithm 1 Modified decoding algorithm for detecting matrix $\boldsymbol{M}$

Input: $M\lambda$, i.e., $h_{\boldsymbol{\lambda}}(\boldsymbol{x})$ for all $\boldsymbol{x} \in \{-1,1\}^{\nu}$.
Output: $\boldsymbol{\lambda}$ or *decoding failure*.
while $h_{\boldsymbol{\lambda}}(\boldsymbol{x}) \neq 0$ **do**
  Compute Fourier coefficients $\hat{h}_{\boldsymbol{a}}$ of $\chi_{\boldsymbol{a}}(\boldsymbol{x})$ in $h_{\boldsymbol{\lambda}}(\boldsymbol{x})$, defined in (16), for all $\boldsymbol{a} \in \{0,1\}^{\nu}$.
  Find a maximal $\tilde{\boldsymbol{a}} \in \{0,1\}^{\nu}$ such that $\hat{h}_{\tilde{\boldsymbol{a}}} \neq 0$.
  Expand $\hat{h}_{\tilde{\boldsymbol{a}}}$ in base d. The coefficients are $\lambda_{\boldsymbol{a},j}$ for $j \in [l_{\boldsymbol{a}}]$, where $l_{\boldsymbol{a}}$ is a non-negative integer that satisfies the condition $d^{l_{\boldsymbol{a}}} \leqslant 2^{|\boldsymbol{a}|-1} < d^{l_{\boldsymbol{a}}+1}$.
  if *The expansion surpasses the maximum integer representable with* $l_{\boldsymbol{a}_i}+1$ *digits in base* d*:* **then**
    Break; declare *decoding failure*.
  end
  Set $h_{\boldsymbol{\lambda}}(\boldsymbol{x}) = h_{\boldsymbol{\lambda}}(\boldsymbol{x}) - \sum_{j\in\{0,1,\cdots,l_{\boldsymbol{a}}\}} \lambda_{\mathbf{a}_i,j} g_{\boldsymbol{a}_i,j}(\boldsymbol{x})$.
  Compute the Fourier coefficient $\hat{h}_{\tilde{\boldsymbol{a}}}$ of $\chi_{\tilde{\boldsymbol{a}}}(\boldsymbol{x})$ in $h_{\boldsymbol{\lambda}}(\boldsymbol{x})$.
  if $\hat{h}_{\tilde{\boldsymbol{a}}} \neq 0$ **then**
    Break; declare *decoding failure*.
  end
end
Return $\boldsymbol{\lambda}$.

Lemma 8 *Consider* $\boldsymbol{M\lambda} = \boldsymbol{v}$*, where* $\boldsymbol{\lambda}$ *is a non-negative integer vector. By enhancing the algorithm described in Bshouty (2009) with an additional step, as provided in Algorithm 1, it can effectively detect a* decoding error *when there exists at least one entry in* $\boldsymbol{\lambda}$ *that is greater than or equal to* d*.*

Proof: Let $h_i(x)$ represent the function remained after iteration i during the recursive decoding procedure described above. Suppose that at iteration i, we select the maximal binary vector denoted as $\boldsymbol{a}_i$. In this case, we can express $h_i(x)$ as: $h_i(x) = h_{i-1}(x) - \sum_{j\in\{0,1,\cdots,l_{\boldsymbol{a}}\}} \lambda_{\mathbf{a}_i,j} g_{\boldsymbol{a}_i,j}$. This means

that all the coefficients corresponding to the functions indexed by $\boldsymbol{a}_i$ are effectively subtracted from $h_i(x)$, resulting in the removal of all such functions, i.e., $g_{\boldsymbol{a}_i,j}$ for all $j \in [l_{\boldsymbol{a}_i}]$. Consequently, the Fourier coefficient of $\chi_{\boldsymbol{a}_i}$ in $h(x)$ becomes zero.

In the event that any of the $\lambda_{\mathbf{a}_i,j}$ values is greater than or equal to d, two scenarios arise. Either the Fourier coefficient of $\chi_{\boldsymbol{a}_i}$ in $h_i(x)$ surpasses the maximum integer representable with $l_{\boldsymbol{a}_i}+1$ digits in base d, or at least one of the $\lambda_{\boldsymbol{a}_i,j}$ values is incorrectly recovered during the decoding process at iteration i. In the former case, the decoder effortlessly detects that at least one entry is not an integer less than d. In the latter case, where a *decoding failure* is not immediately apparent from the Fourier coefficient of $\chi_{\boldsymbol{a}_i}$ in $h_i(x)$, the decoder must additionally verify whether the Fourier coefficient of $\chi_{\boldsymbol{a}_i}$ in the function derived from the *pruning* of $h_i(x)$ is zero or not. If this coefficient is indeed zero, it shows that the entries are determined correctly, implying that all such entries are integers less than d. Conversely, if the coefficient is non-zero, the decoder realizes that at least one entry violates the initial assumption. ■

The result of Lemma 8 implies that by incorporating this modification into the decoder for the d-detecting matrix constructed in Bshouty (2009), the decoder gains the ability to determine whether the input vector adheres to the assumption that all of its non-negative integer entries are less than d or not. It can also correctly recover these entries if they do satisfy the assumption. Later in this section, we will demonstrate how this modification enables our decoding algorithm for the (n,k)-QGT problem to identify cases where it cannot recover the underlying incident vector. This capability allows the algorithm to declare a *decoding failure*, affirming that if it does produce a vector, that vector must indeed be the unique incident vector.

4.2. (n,k)-Search matrix with $k = O(\log n)$

In this subsection, we focus on addressing the (n,k)-QGT problem with a specific condition: when k is in the order of $O(\log n)$. This scenario represents a much sparser setting compared to the case where $k = n^{\alpha}$. Note that the number of defective items still grows large, although with a speed that is logarithmic in n, which means that algorithms designed for the *sparse* regime might not provide guaranteed solutions. To tackle this challenge, we leverage the construction of BCH codes from the literature of coding theory.

An $[n,k,d]$ binary code $\mathcal{C}$ is a k-dimensional linear subspace of the binary vector space of dimension n over $\mathbb{F}_2$. The rate of $\mathcal{C}$ is $R \stackrel{\text{def}}{=} \frac{k}{n}$. The parameter d represents the minimum Hamming distance between the codewords. A generator matrix of $\mathcal{C}$ is a $k \times n$ matrix whose rows span $\mathcal{C}$. A parity-check matrix of $\mathcal{C}$ is an $(n-k) \times n$ whose rows span the null space of G, i.e., $\boldsymbol{G}\boldsymbol{H}^T = 0$. This implies that $\boldsymbol{H}\boldsymbol{x} = 0$ if $\boldsymbol{x} \in \mathcal{C}$. A code with a minimum distance of d can uniquely recover any codeword $\boldsymbol{x}$ in the presence of up to $t = \left\lfloor \frac{d-1}{2} \right\rfloor$ errors. This property implies that all $\mathbf{H}\boldsymbol{e}$'s are distinct for every $\boldsymbol{e} \in \{0,1\}^n$ with Hamming weight at most t.

Note that all arithmetic operations are conducted within the field $\mathbb{F}_2$, which also implies validity over the real numbers $\mathbb{R}$. Consequently, a parity-check matrix of a t-error correcting code satisfies the criteria of an (n,t)-search matrix, as outlined in Definition 1.

In particular, we employ binary BCH codes, known for providing an optimal trade-off between minimum distance and code rate for cases where $d \leqslant \frac{n}{\log n}$. Such codes has been also utilized as a building block to construct test matrices for the QGT problem in Karimi et al. (2019).

Let $n = 2^m - 1$ for for some $m \geqslant 3$ and $t < 2^{m-1}$ be an integer. There exists a binary BCH that corrects t errors with $n - k \leqslant mt$ and $d \geqslant 2t + 1$. Let

$$\tilde{\boldsymbol{H}} \stackrel{\text{def}}{=} \begin{bmatrix} 1 & \gamma & \gamma^2 & \cdots & \gamma^{n-1} \\ 1 & \gamma^3 & (\gamma^3)^2 & \cdots & (\gamma^3)^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \gamma^{2t-1} & (\gamma^{2t-1})^2 & \cdots & (\gamma^{2t-1})^{n-1} \end{bmatrix}, \tag{17}$$

where γ is a *primitive* element in $\mathbb{F}_{2^m}$. Then, the matrix $\boldsymbol{H}$ that is derived from $\tilde{\boldsymbol{H}}$ by replacing its entries by their corresponding representation in $\mathbb{F}_2$ as a column vector is a parity-check matrix of this binary code. Since each element of the the extension field $\mathbb{F}_{2^m}$ is represented in m elements in the binary base field, the matrix $\boldsymbol{H}$ has at most $mt = t\log(n+1)$ rows. Equivalently, the matrix H is an (n, t)-search matrix with at most $t\log(n+1)$ measurements/rows. Moreover, a binary vector $\boldsymbol{x}$ of Hamming weight at most t can be recover from $\boldsymbol{Hx}$ in $O(tn)$ time complexity by utilizing the well known algorithms for decoding BCH codes, e.g., Berlekamp-Massey algorithm Berlekamp (2015); Massey (1969) the Euclidean algorithm Sugiyama et al. (1975) the Berlekamp-Welch algorithm Welch and Berlekamp (1986). We utilize the aforementioned parity-check matrix of the BCH code as a search matrix that solves the QGT problem where the number of defective items is logarithmic in the total number of items n.

4.3. Our Construction

Now that all necessary building block and theoretical results are established, we are ready to propose our construction for (n, k)-search matrices. The main idea behind our construction can be described as in the following: The set of all items are first partitioned into $\frac{k}{\ln k}$ groups. Then, the result of Corollary 7 implies that there are at most $e \ln k$ in each partition with probability $1 - o(1)$. The problem then can be solved by using the concatenation scheme outlined in 2, where the (n, d)-detecting matrix reviewed in 4.1 and the parity-check of a BCH code with suitable parameters, as discussed in 4.2 are used as building blocks.

Let $\boldsymbol{D}$ denote a $(\frac{k}{\ln k}, (e+\gamma)\ln k)$-detecting matrix constructed according to the scheme overviewed in 4.1 for some $\gamma > 0$. Let $\tilde{\boldsymbol{H}}$ denote the parity-check matrix of a $(\tilde{n}, (e+\gamma)\ln k)$-BCH code, where $\tilde{n}$ is the smallest power of 2 such that $\frac{n \ln k}{k} \leqslant \tilde{n}$. Let also $\boldsymbol{H}$ be a submatrix of $\tilde{\boldsymbol{H}}$ consists of any $\frac{k}{\ln k}$ columns of it. One can see that $\boldsymbol{H}$ is a $(\frac{k}{\ln k}, (e+\gamma)\ln k)$-search matrix since $\tilde{\boldsymbol{H}}$ is a $(\tilde{n}, (e+\gamma)\ln k)$-search matrix. Then, the following matrix is our proposed test matrix for the (n, k)-QGT problem in the sublinear regime where $k = n^{\alpha}$ for some $0 < \alpha < 1$:

$$\boldsymbol{T} = \boldsymbol{D} \otimes \boldsymbol{H}. \tag{18}$$

The next theorem establishes the result on the successfully decoding the underlying incident vector when using this test matrix.

Theorem 9 *Consider a binary vector $\boldsymbol{x}$ sampled from the uniform distribution over $\mathcal{B}_{n,k}$. Then, $\boldsymbol{x}$ can be recovered from $\boldsymbol{Tx}$, where $\boldsymbol{T}$ is characterized in (18), with probability $1 - o(1)$ and the decoding complexity of*

$$O(k^2 + n\log^2 k),$$

using no more than

$$(2e+\gamma)\frac{k}{\log k}\log(\frac{n}{k})\log\log k,$$

for any $\gamma > 0$. Moreover, in cases where the decoder cannot obtain a unique reconstruction of $\boldsymbol{x}$, it can reliably detect this condition and report a decoding failure.

Proof: The result of Corollary 7 implies that the vector $\boldsymbol{x}$ is $(\frac{k}{\ln k}, (e+\gamma)\ln k)$-balanced with probability $1-o(1)$, for any constant $\gamma > 0$. In other words, for any positive constant γ, it is *highly likely* that the number of defective items allocated to each of the $\frac{k}{\ln k}$ partitions will not exceed $(e+\gamma)\ln k$. Therefore, the result of Theorem 4 can be applied to this case with $n_1 = \frac{k}{\ln k}, m_1 = \frac{2n_1}{\log n_1}\log((e+\gamma)\ln k)$ according to (14), and $n_2 = \frac{n\ln k}{k}, m_2 = (e+\gamma)\ln k \log(\tilde{n}+1)$, as described in 4.2, and $d = (e+\gamma)\ln k$. Recall that $\tilde{n}$ is the smallest power of 2 such that $\frac{n\ln k}{k} \leqslant \tilde{n}$. This implies $\tilde{n} \leqslant \frac{2n\ln k}{k}$, and thus, $m_2 \leqslant (e+\gamma)\ln k \log(\frac{2n\ln k}{k}+1)$. Specifically, the result of Theorem 4 implies that the underlying incident vector can be recovered with computational complexity of $m_2 O_{\mathscr{A}} + n_1 O_{\mathscr{B}}$, where $O_{\mathscr{A}}$ and $O_{\mathscr{B}}$ denote the computational complexities of the underlying decoders associated with $\boldsymbol{D}$ and $\boldsymbol{H}$, respectively. Recall that the computational complexity of the decoding algorithm described in 4.1 is $O_{\mathscr{A}} = O(n_1^2) = O(\frac{k^2}{\ln^2 k})$ and the decoding complexity of BCH codes, described in 4.2, is $O_{\mathscr{B}} = O(m_2 n_2) = O(\frac{n\ln^3 k}{k})$. Therefore, the overall combined computational complexity is $O(k^2 + n\log^2 k)$Note that decoding failure only occurs when $\boldsymbol{x}$ is not $(\frac{k}{\ln k}, (e+\gamma)\ln k)$-balanced, in which case the decoder detects it during the decoding procedure (Lemma 8). Also, the number of rows in $\boldsymbol{D}$ is upper bounded by $\frac{2n_1\log d}{\log n_1}(1+o(1)) = \frac{2k}{\log k \ln k}\log\log k\,(1+o(1))$, according to (14), and the number of rows in $\boldsymbol{H}$ is $m_2\log(\tilde{n}+1) \leqslant (e+\gamma)\ln k\log(\frac{2n\ln k}{k})$. Overall, the number of rows in $\boldsymbol{T}$ is upper bounded by $(2e+\gamma)\frac{k}{\log k}\log(\frac{n}{k})\log\log k$, for an arbitrarily small $\gamma > 0$. ■

Note that the constructions for $\boldsymbol{T}$ with efficient decoding schemes in the literature (Soleymani et al. (2023); Gebhard et al. (2022); Karimi et al. (2019)) still fall short of the information-theoretic bound by a factor of $\log k$. Recently, Hahn-Klimroth and Müller (2022) closed the gap by providing a construction with $O(n^3)$ decoding complexity. Our result falls between these two lines of works; our algorithm narrows the gap from $\log k$ to $\log\log k$, presenting an order-wise improvement over the sub-optimal existing work in terms of the number of tests. Additionally, our algorithm offers an order-wise improvement in decoding complexity compared to the recently proposed asymptotically optimal construction. In other words, our construction can be viewed as a scheme that achieves near-linear decoding complexity for $\alpha < \frac{1}{2}$ at the expense of deviating by a factor of $\log\log k$ from the asymptotic bound for the number of tests However, despite a double logarithmic gap, our scheme numerically outperforms the asymptotically optimal construction when α is in the higher range of the unit interval for moderate values of k. The existing optimal solution requires fewer tests only when k is superficially large. Specifically, our scheme requires $(e\log\log k)m_0$ tests, while the asymptotically optimal one needs $(\frac{1+\sqrt{\alpha}}{1-\sqrt{\alpha}})m_0$, where m_0 is the information theoretic lower bound. The crossover value for k is $k^* = \exp(\exp(\frac{1+\sqrt{\alpha}}{e(1-\sqrt{\alpha})}))$, growing double-exponentially in $\frac{1}{1-\sqrt{\alpha}}$. Hence, as α approaches 1, the crossover becomes superficially large, highlighting the suitability of our scheme in the sub-linear regime for higher α values. For example, with $\alpha = 0.7$, we find $k^* \approx 10^{27}$.

References

Elwyn R Berlekamp. *Algebraic coding theory (revised edition)*. World Scientific, 2015.

Nader H Bshouty. Optimal algorithms for the coin weighing problem with a spring scale. In *COLT*, volume 2009, page 82, 2009.

Emmanuel J Candès, Justin Romberg, and Terence Tao. Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. *IEEE Transactions on information theory*, 52(2):489–509, 2006.

Chang-Chang Cao, Cheng Li, and Xiao Sun. Quantitative group testing-based overlapping pool sequencing to identify rare variant carriers. *BMC bioinformatics*, 15(1):1–14, 2014.

Gianluca De Marco, Tomasz Jurdziński, and Dariusz R Kowalski. Optimal channel utilization with limited feedback. *Journal of Computer and System Sciences*, 119:21–33, 2021.

David L Donoho. Compressed sensing. *IEEE Transactions on information theory*, 52(4):1289–1306, 2006.

Uriel Feige and Amir Lellouche. Quantitative group testing and the rank of random matrices. *arXiv preprint arXiv:2006.09074*, 2020.

Oliver Gebhard, Max Hahn-Klimroth, Dominik Kaaser, and Philipp Loick. On the parallel reconstruction from pooled data. In *2022 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 425–435. IEEE, 2022.

Max Hahn-Klimroth and Noela Müller. Near optimal efficient decoding from pooled data. In *Conference on Learning Theory*, pages 3395–3409. PMLR, 2022.

Esmaeil Karimi, Fatemeh Kazemi, Anoosheh Heidarzadeh, Krishna R Narayanan, and Alex Sprintson. Sparse graph codes for non-adaptive quantitative group testing. In *2019 IEEE Information Theory Workshop (ITW)*, pages 1–5. IEEE, 2019.

James Massey. Shift-register synthesis and bch decoding. *IEEE transactions on Information Theory*, 15(1):122–127, 1969.

C. J. Park. Random allocations (valentin f. kolchin, boris a. sevast'yanov and vladimir p. chistyakov). *SIAM Review*, 22(1):104–104, 1980. doi: 10.1137/1022018. URL https://doi.org/10.1137/1022018.

Martin Raab and Angelika Steger. "balls into bins"—a simple and tight analysis. In *International Workshop on Randomization and Approximation Techniques in Computer Science*, pages 159–170. Springer, 1998.

Mahdi Soleymani, Hessam Mahdavifar, and Tara Javidi. Non-adaptive quantitative group testing via plotkin-type constructions. In *2023 IEEE International Symposium on Information Theory (ISIT)*, pages 1854–1859. IEEE, 2023.

DA Sprott. Urn models and their application—an approach to modern discrete probability theory, 1978.

Yasuo Sugiyama, Masao Kasahara, Shigeichi Hirasawa, and Toshihiko Namekawa. A method for solving key equation for decoding goppa codes. *Information and Control*, 27(1):87–99, 1975.

Chao Wang, Qing Zhao, and Chen-Nee Chuah. Group testing under sum observations for heavy hitter detection. In *2015 Information Theory and Applications Workshop (ITA)*, pages 149–153. IEEE, 2015.

L. R. Welch and E. R. Berlekamp. Error correction for algebraic block codes. *U.S. patent no. 4,633,470*, Dec. 30, 1986.

Appendix A. Proof of Lemma 6

proof with figure

Appendix B. Summary of the Construction for Detecting Matrix in Bshouty (2009)

In this section, we first introduce a construction for an $(k_1, k_2, \cdots, k_n)$-*detection* matrix, i.e., a test matrix for the coin weighing problem with constraints that achieves the asymptotically optimal number of tests. This scheme is proposed by Bshouty (2009). The ith component of a vector $\boldsymbol{x}$ is denoted by $\boldsymbol{x}_i$. Let $<$ denotes the usual lattice partial ordering over on dimensional vectors. Specifically, for two $\boldsymbol{a}, \boldsymbol{b} \in \{0,1\}^n$, we say $\boldsymbol{b} < \boldsymbol{a}$ if and only if the support of $\boldsymbol{b}$ is a subset of $\boldsymbol{a}$. For $\boldsymbol{a}, \boldsymbol{b} \in \{0,1\}^n$ where $\boldsymbol{b} < \boldsymbol{a}$, we define

$$f_{\boldsymbol{a},\boldsymbol{b}}(\boldsymbol{x}) = \prod_{\boldsymbol{a}_i=1} \frac{(-1)^{\boldsymbol{b}_i}\boldsymbol{x}_i + 1}{2}. \tag{19}$$

Algorithm 2 provides the steps of the construction proposed for detection matrices in Bshouty (2009) in detail. The algorithm outputs a binary matrix $\boldsymbol{M}$ whose number of rows are asymptotically optimal as the number of coins, namely l, grows large.

Algorithm 2 Constructing an $(k_1, \cdots, k_l)$ detecting matrix $\boldsymbol{M}$.

Input: $k_1, \cdots, k_l$, $(k_1 \leqslant k_2 \cdots \leqslant k_l)$, l.
Find the maximal integer ν such that

$$(\nu - 2)2^{\nu-1} \leqslant \log(k_{l-2^\nu}^{2^\nu} \prod_{i=1}^{n-2^\nu} k_i). \tag{20}$$

Set $\boldsymbol{M}_{2^\nu \times l} = 0_{2^\nu \times l}$, $r = 0$ and $s = 0$.
for $\forall \boldsymbol{a} \in \{0,1\}^\nu \backslash \{0\}$ **do**
  Find $l_{\boldsymbol{a}}$ such that

$$k_{r+1}k_{r+2}\cdots k_{r+l_{\boldsymbol{a}}} \leqslant 2^{\|\boldsymbol{a}\|_0 - 1} < k_{r+1}k_{r+2}\cdots k_{r+l_{\boldsymbol{a}}}k_{r+l_{\boldsymbol{a}}+1}$$

  Construct $G_{\boldsymbol{a}} = \{f_{\boldsymbol{a},\boldsymbol{b}(x)} : \boldsymbol{b} < \boldsymbol{a}, \|\boldsymbol{b}\|_0 \equiv 0 \mod 2\}$ (check $\|G_{\boldsymbol{a}}\|_0 = 2^{\|\boldsymbol{a}\|_0 - 1}$)
  Choose any collection of subsets

$$G_{\boldsymbol{a},0}, G_{\boldsymbol{a},1}, \cdots, G_{\boldsymbol{a},l_{\boldsymbol{a}}} \subset G_{\boldsymbol{a}}$$

  such that

$$\|G_{\boldsymbol{a},0}\|_0 = 1, \quad \|G_{a,i}\| = k_{r+1}k_{r+2}\cdots k_{r+i} \quad \forall i = 1, \cdots l_{\boldsymbol{a}}.$$

  for $i \in [l_{\boldsymbol{a}}]$ **do**
    Set

$$h_{\boldsymbol{a},i}(\boldsymbol{x}) \stackrel{\text{def}}{=} \sum_{g(\boldsymbol{x}) \in G_{\boldsymbol{a},i}(\boldsymbol{x})} g(\boldsymbol{x}).$$

    Set

$$\boldsymbol{M}[:, r+i] = h_{\boldsymbol{a},i}(\boldsymbol{x})\big|_{\{-1,+1\}^\nu}.$$

  end
  Set

$$h_{\boldsymbol{a},l_{\boldsymbol{a}}}(\boldsymbol{x}) \stackrel{\text{def}}{=} \sum_{g(\boldsymbol{x}) \in G_{\boldsymbol{a},l_{\boldsymbol{a}}}(\boldsymbol{x})} g(\boldsymbol{x}).$$

  Set

$$\boldsymbol{M}[:, l - 2^\nu + s + 1] = h_{\boldsymbol{a},l_{\boldsymbol{a}}}(\boldsymbol{x})\big|_{\{-1,+1\}^\nu}.$$

  Set $r = r + l_{\boldsymbol{a}}$, and, $s = s + 1$.
end
Return $\boldsymbol{M}$.

Let m denote the number of rows in $\boldsymbol{M}$, and $(k_1, \cdots, k_l)$-detection matrix constructed according to Algorithm 2. It is proved in Bshouty (2009) that $m = \frac{2l \log \frac{k}{l}}{\log l}(1 + o(1))$.